BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 — A1

(11) International Publication Number: **WO 98/54662**

(43) International Publication Date: 3 December 1998 (03.12.98)

(21) International Application Number: PCT/US98/10679

(22) International Filing Date: 26 May 1998 (26.05.98)

(30) Priority Data:
08/863,680 27 May 1997 (27.05.97) US

(71) Applicant: ARKONA, INC. [US/US]; Suite 3409, 4505 South Wasatch Boulevard, Salt Lake City, UT 84124 (US).

(72) Inventors: ZOLLINGER, John, M.; 3563 East Wasatch Grove Lane, Salt Lake City, UT 84121 (US). DEVINE, Johnathan; Loft 408, 300 Beale Street, San Francisco, CA 94105 (US).

(74) Agents: STRINGHAM, John, C. et al.; Workman, Nydegger & Seeley, 1000 Eagle Gate Tower, 60 East South Temple, Salt Lake city, UT 84111 (US).

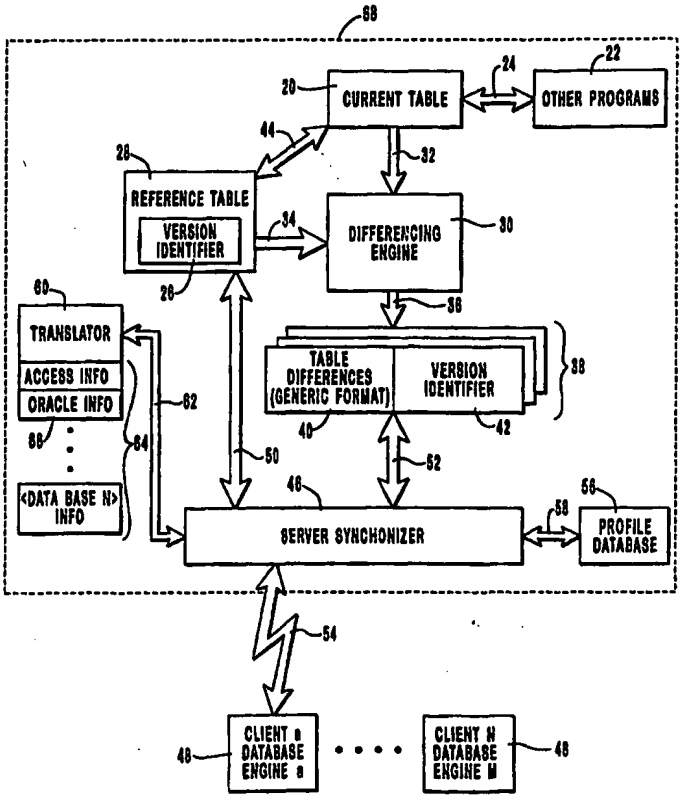
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD, COMPUTER PROGRAM PRODUCT, AND SYSTEM FOR DISTRIBUTING CHANGES MADE IN A DATA STORE TO REMOTE CLIENT COPIES OF THE DATA STORE

(57) Abstract

A method, computer program product, and system that allows changes made to an original database table found on a server computer to be reflected in client copies of the database table based on intermittent client requests for synchronization. A server makes periodic updates of table differences between current table (20) receiving database change events and reference table (28). Each client copy of a database table and update (created by the server has a sequential version number associated therewith). The server will compare the version number of a client copy of a database table with the most recent version number of the table on the server to determine which updates need be applied in order to make the client copy current. Next, the updates will be translated from a generic format into instructions that are specific to the type of database engine being run on the client. Finally, the instructions are transmitted to the client (along with the new version number) so that the client may operate the database engine to apply the instructions for making the database table current with the original managed on the server.



METHOD, COMPUTER PROGRAM PRODUCT, AND SYSTEM FOR DISTRIBUTING CHANGES MADE IN A DATA STORE TO REMOTE CLIENT COPIES OF THE DATA STORE

BACKGROUND OF THE INVENTION

1. The Field of the Invention

The field of the present invention pertains to distributing changes made to a database, database table, or other data store on a server computer out to read-only copies of the data store found on one or more client computers. More specifically, the invention deals with distributing such database changes in a manner that efficiently uses system resources and is quickly achieved. Another area of the present invention pertains to client systems that are intermittently (as apposed to continuously) connected to a server system requiring communication and synchronization of information on both systems.

2. Present State of the Art

In many situations, it is desirable to distribute an original database, database table, or other data store on a server computer to one or more client computers at various locations. Furthermore, when the original data store at the server is changed in some way (*e.g.*, the addition, deletion, or modification of a record) it is desirable to distribute those changes out to the various client copies of the data store or database table so that the client copies may be current with the original.

A data store is any form of information readable with the assistance of a general purpose computer. The most common type of data store are traditional databases but any form of data storage may require that changes made to an original data store on server to be distributed outward to client copies of that data store. For illustration purposes, a database table is used throughout as an example of a data store, though many other kinds of data store exist.

The client copies of the original data store or elements thereof such as a database table are in one respect read-only copies since any changes made by the client will not be distributed back to the original. This differentiates the present area of the invention from the art of data replication wherein a change made to any copy of the database must be replicated at every other database or database table.

The usefulness of information distribution from an original data store to client representation of the data store is manifest in applications where the client is a remote laptop computer that is only intermittently connected for brief periods of time with the centralized server computer. The client copy of the database information may be used

as exists on the server side. This attribute and presumption can make deployment of such systems costly by requiring the purchase of a specific type of database engine for every client using the system. Furthermore, the original database tables or databases desired for remote distribution may be managed by many different database engines thereby requiring each client to use or maintain multiple database engines.

It would therefore be an advance in the art to allow client representations of a data store to be managed by a different type of data store engine than that managing the server data store. This would allow a single data store engine to be found on each client that could handle multiple data stores, such as databases or collections of documents, that are originally created and managed on the server by different types of data store engines. Furthermore, existing data store and database engines found on a particular client system may be leveraged without necessitating the purchase of new or different data store engines in order to integrate with a system of distributing copies of a data store, such as a database table, as described previously.

SUMMARY OF THE INVENTION

The present invention quickly delivers database changes made to an original database table on a server to a requesting client so that the client may apply the differences to make the client copy of the database table current.

The present invention creates and stores difference updates that can be used for quickly sending database table differences to a client for use in making a client copy of a database table current.

In addition, the present invention translates database changes to instructions that can be understood by a particular type of database engine residing on a client computer thereby allowing the client to update the client copy of a particular database table in order to make it current.

The present invention provides client copies of database tables to be managed by different database engines and yet contain the same data and the same general organization.

Furthermore, the present invention allows database changes to be made to a data store located on a server to be distributed out to client copies of the data store in an efficient and timely manner.

Additional advantages of the invention will be set forth in the description which follows, and in part will be obvious from the description, or may be learned by the practice of the invention. The advantages of the invention may be realized and obtained by means of the instruments and combinations particularly pointed out in the appended claims.

These and other features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

In order that the manner in which the above-recited invention and other advantages of the invention are obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

Figure 1 is a block diagram illustrating the architecture of a system implementing the method of the present invention wherein a server synchronizer component will communicate intermittently with one or more clients in order to distribute changes made to a data base table on the server out to the respective client copy of the database table upon client request;

Figures 2A-2D are diagrams showing the state of an example database table of employee information at four different moments in time;

Figures 3A-3B are diagrams showing the contents of two particular updates with Figure 3A showing the changes that occurred between Figure 2A and Figure 2B while Figure 3B shows the changes that occurred between Figure 2B and 2C;

Figure 4 is a block diagram showing the state of progression of Figures 2A-2D correlated with the changes represented in the updates shown in Figures 3A-3B;

Figure 5 is a flow chart showing the processing steps taken by the differencing engine of Figure 1 to create an update of differences between the current state of a database table and a reference copy of the database table that may be used in generating and distributing database table differences to clients having client copies of the database table;

Figure 6 is a flow chart showing the processing steps taken by the server synchronizer component of Figure 1 in order to distribute the appropriate database table differences to a requesting client according to the present invention;

Figure 7 is a flow chart showing the processing steps taken by a client in order to request and receive the correct differences from a server that may be applied to the client copy of the database table in order to make it current.

or in different parts of the same media. For example, a storage means comprising RAM and disk storage could be logically partitioned so that item A is stored in a portion of RAM (first partition), item B is stored in another portion of RAM (second partition), and item C is stored on disk (third partition).

As used herein, the term "database" or "data store" refers to any collection of information that can be read or accessed by a program running on a general purpose computer. While this definition entails standard database formats, such as SQL databases, it also contemplates other entities such as computer files that may have any form of data contained thereon, or collections of files. For example, a set of documents, each document being a file in the format of a standard word processor, would constitute a data store. Furthermore, the data within a file or traditional database is unlimited as to its meaning. In other words, the data could be sound data, video images, statistical information, etc.

As used herein, the term "database table" or "table" refers to the row/column organization of data in a standard SQL database. Again, the cells or elements of a database table may contain data or information that is unlimited in its nature. Data sources can also organize information in entity, attribute, and relationship form (in addition to other forms).

As used herein, the term "database engine" or "data store engine" refers to a software program that can understand and interact with a particular data store. Such a database engine would include varieties of SQL database engines, such as Microsoft® Access™, or Borland® Paradox™; as well as word processors, such as Microsoft® Word, and other programs that may read or organize computer information. Traditional data source types include, but are not limited to the following: relational, hierarchal, object-relational, object oriented, flat files, etc. Furthermore, a database engine must be able to process database instructions in order to change database contents and may consist of multiple software components acting in harmony one with another. A data source type simply identifies a particular class or implementation of an engine such as a Microsoft® Access™ SQL database engine.

As used herein, a "database change event" is anything that changes the state of a database, such as additions, deletions, or modification of records. Furthermore, other types of events may make changes to a database including, by way of example and not limitation, sorting a database, adding an extra field or column to a database table, changing "metadata" parameters such as passwords, permissions, logins, structure, etc.

As used herein, the term "update" refers to a set of differences on a particular database taken between two separate states of that database or database table. Generally,

synchronization. One or more clients, illustrated by the series of clients 48a-48n may utilize the services of the server synchronizer component 46 and have contained thereon a copy of the database table.

The block diagram of Figure 1 illustrates the invention for a single database table for purposes of teaching the present invention and that actual implementations will likely have many different database tables with each client "subscribing" to one or more of the database tables. It should also be noted that the present invention extends beyond a database table and can be used for any form of database or stored information that would be distributed out to clients in read-only fashion. The invention applies particularly to clients that are only intermittently connected to the server synchronizer component 46.

One example of an intermittent connection environment would be the servicing example explained previously. In that environment, a parts database is centrally managed and updated but is used by field service representatives having laptop computers (*i.e.*, clients). The field service representatives will only intermittently connect with the home office server computer on a periodic and often random basis ranging from a couple of times per day to weekly or even less frequently.

Referring back to Figure 1, note that the client computers will not necessarily change the data in the client copies of the database table though this may occur in some circumstances. If such changes are made to the client copy of the database tables by the client, the changes will not be propagated back to the original table managed on the server computer and could actually be lost when update instructions are received by one of the clients 48a-48n.

The server synchronizer component 46 has access to the reference table 28 as represented by arrow 50 in order to transfer or copy the reference table 28 onto a respective client in the series of clients 48a-48n. Also, the server synchronizer component 46 will communicate with the series of updates 38 as represented by arrow 52 in order to use those updates in synchronizing the client copy of the database table located on a respective client system with the original database found on the server.

The intermittent connection between the server, being represented by all the components encircled by the dashed line 68, and each of the series of clients 48a-48n is represented by arrow 54. The nature of the communication represented by arrow 54 in a currently preferred embodiment is a direct modem connection, however, any communications network or method (*i.e.*, by way of a disk or tape) may be used so that the communication path between client and server may be made. Furthermore, the logical connection (*i.e.*, the actual contact between the client server or handshaking) may

these parameters in the synchronization request. Alternatively, the client may simply identify itself and all such information may be stored in the corresponding client entry of the profile database 56 that may be accessed by the server synchronizer component 46.

In either case, the server synchronizer component 46 will know or be able to ascertain the status of the client copy of the database table in order to determine which updates of the series of updates 38 need to be applied to that table in order to make it current. The server synchronizer component 46 will also be able to deliver the information that the translator component 60 will need in order to translate the table differences 40 from a generic format to the correct format or instructions for the type of database engine on the particular client requesting synchronization.

Generally, it is preferred to push as much information up to the centralized server as possible so that a client component that interfaces with the server has as little sophistication as possible. In other words, the client will simply receive instructions from the server that may be given to a database engine in order to apply the relevant updates to the client copy of the database table. In such a minimal implementation, the client component need only store its identifying information for communicating and identifying itself to the server synchronizer component 46. Additionally, minimal client software will have the ability to communicate with its native database engine, though client software may be so written that the same client code may be configured to interface with a variety of different types of database engines. Such an arrangement allows the client component to be very flexible when adding new types of database engines supported by the current system shown in Figure 1.

Referring to Figures 2A-2D, different states of a database table having employee information are shown with each state at a different point in time and progressing sequentially in time from Figure 2A to Figure 2D. For illustration purposes, the database table is small both in terms of columns and rows and those skilled in the art will appreciate that a database table of any size may be used according to the concepts illustrated in the present invention. Furthermore, any form of database used interchangably in place of the database table is considered within the scope of the present invention, including but not limited to such things as data files, other databases organized other than row-column format, etc.

Referring to Figures 3A-3B, two updates organized in an arbitrary generic format are shown. Again, the format is chosen for illustration purposes only and those skilled in the art will appreciate that many different formats or conventions may be chosen. Specifically, Figure 3A corresponds to the changes between the database table that occurred going from the state in Figure 2A to the state in Figure 2B. In other words, the

manually, automatically after so many updates have been made, based on significant structural changes made to the table requiring a re-copy of the table to the client, or any other relevant criteria. Typically, a major revision is required when the entire database table should be copied to a client such as at initial creation of the table or when the overhead of applying the many updates is greater than simply copying the table.

If a minor revision is determined at step 78, as in the case between the database table states shown in Figures 2A and 2B, the versioning is incremented at step 80 for a minor revision. In a currently preferred versioning system, major and minor revisions are separated by a decimal point, therefore in the current example, the version marker would increment from 1.0 to 1.1.

The differences are generated between the current table 20 and the reference table 28 by the differencing engine 30 and stored as an update in the series of updates 38 (*See* Figure 1). These differences are prepared as part of an update (*e.g.*, update 1.1 shown in Figure 3A).

Between the state of the database table in Figure 2A and Figure 2B, three changes were made. Namely, the employee in row one became married, a new employee was added (Mr. Mauss), and a former employee deleted at row 2 (Mr. Presley). In Figure 3A, these changes are stored in an arbitrary generic format with a change-type indicator separated by a ":" followed by a location field separated by a "=>" followed by the data of the change itself. For modifications to existing table cells, the change-type indicator is signified by a "M," the location of the cell is given by the row number and the column number, and the data is the new cell data. For additions of a new record or row, the change-type indicator is "A," the location field indicates after which row the new record should be inserted, and the data field indicates all the cells therein. Finally, a deletion will be signified by a "D" change-type indicator and the location field contains the row number to be deleted (no data is associated with a delete).

Once the differences have been generated at step 82, the differences are stored in generic format and the current version number (in this case 1.1) is associated with the difference update at step 84. Finally, the current table 20 (Figure 2B) is copied to the reference table 28 (now also Figure 2B) to complete the update sequence. Note that the current version number (at this point 1.1) is used to indicate both the newly copied reference table 28 as well as the update just created. Semantically, version 1.0 of the table having the update 1.1 applied thereto would be the same as version 1.1 of the table.

At the end of the update sequence, the current table goes back to receiving database change events at step 74 until another update creation is initiated at step 76. The same process will occur for creating update 1.2 as shown in Figure 3B as was

explained previously. For example, a client could simply ask for all updates since a certain date without tracking which version number it actually has. The server could compare the date to a file creation dates for the updates (if stored in a file) or other date and time stamp information in order to assure the correct updates are used to make the client copy of the database table current.

Referring to the flow chart of Figure 6, the processing steps necessarily taken by the server computer for implementing the present invention are shown. At step 94, difference updates are created with version numbers for all database tables in the system. Each database table will have client copies thereof on one or more of the various clients to the system. Furthermore such updates are stored in a generic format which may later be translated to database engine instructions destined for database engine types found on the appropriate client requesting synchronization. Typically, such updates are handled by the differencing engine 30 as shown in Figure 1.

Next, the server synchronizing component 46 receives a synchronization request from a client computer at step 96. In one embodiment of the invention, a remote client will dial into the server computer using a modem and phone line as a communications network and "login" or otherwise identify itself and begin a session with the server computer.

The synchronization request is validated at step 98 making access to profile information in the profile database 56 as shown in Figure 1. This is a security feature that assures that a valid client is receiving or requesting information from the server. Furthermore, the profile database information may also include reference to new database tables assigned to the client. The server synchronizer component 46 would then be required to communicate the new tables to a particular client at a later point in time as will be shown hereafter.

At step 100, the server synchronizer component 46 will determine which database tables are applicable to the client making the request. This information may be presented directly by the client itself in the request or references to applicable database tables may be stored in the profile information pertinent to that particular client. In either instance, the server synchronizer component 46 will be able to determine the appropriate database tables in step 100 and those skilled in the art will see many schemes and methods by which this may be accomplished.

At step 102, the state of the existing client copy of the database tables and their particular version number are determined, again this information may be provided in the request from the client or may be centrally stored in the profile database 56 or other area

Referring now to Figure 7, a flow chart is shown illustrating the processing steps taken on the client computer for synchronizing the client copy of one or more database tables with the originals of the same found on the server computer. At step 112, the client will establish a connection with the server computer. This may entail a modem to modem connection over telephone line or some other means as was explained previously.

The client will generate and send a request for synchronization to the server computer to be received by the synchronization component 46. The synchronization request will at the very least contain information identifying the client and may contain information regarding the client copies of database tables existing at the client along with associated version numbers, the type of database engine being run at the client, password information, etc.

Once the synchronization request is sent, the client will wait until receiving instructions and the current version number(s) from the server for updating the client copies of each database table contained thereon at step 116. These instructions will be of the appropriate format for the native database engine type found on the client.

Finally, at step 118, the client will operate the database engine and the instructions received previously at step 116, to apply the difference updates and/or copy new tables from the server in order to make each database table current at the client. The client can thus be made in a flexible manner to operate with many different types of database engines without necessarily involving a large amount of redeployment effort at the client. On the other hand, a more sophisticated redeployment effort takes place at the server in order to support the new or different database engine type.

It is apparent that complex and flexible systems may be created using the present invention. With respect to the field service representative example explained previously, a server may hold technical documents in Microsoft® Word™ format (one data store) and a customer database in a SQL database table using a Microsoft® Access® database engine. The client, on the other hand, may manage the client copy of the customer database using a Borland® Paradox® database engine and the client copy of the technical documents using Corel® WordPerfect® or Folio® Infobase® as the appropriate database engine. In the above-mentioned scenario, translation will allow the database change events incorporated as differences in one or more updates to be reflected in instructions of the proper format. Note that in the above example a single client will receive instructions pertaining to two different database engine types.

Those skilled in the art will also see the ability to auto-update the client portion itself so that if a new database engine type is presented or made known to the server, at that point, the server may download a new addition of the client code that will interact

1. A method of distributing database differences corresponding to change events made to a data store located on a server computer to client copies of the data store located on one or more client computers comprising the steps of:

creating and storing on the server computer one or more versioned updates, each update containing data store differences corresponding to data store change events made to the data store;

determining which updates are necessary for making the client copy of the data store current; and

generating and transmitting the specific data store differences to the client computer based upon the necessary updates.

2. A method of distributing database differences corresponding to change events made to a database table located on a server computer to client copies of the database table located on one or more client computers comprising the steps of:

creating and storing on the server computer one or more sequentially versioned updates, each update containing database differences corresponding to database change events made to the database table since the preceding update;

receiving, from a client computer, a request for all the database differences needed to make the client copy of the database table current;

determining which updates are necessary for making the client copy of the database table current; and

generating and transmitting the specific database differences to the client computer based upon the necessary updates, whereby the specific database differences are used by the client computer to make the client copy of the database table current.

3. A method as recited in claim 2 wherein the client computer supplies the version of the client copy of the database table as part of the request and the determination the necessary updates is done by comparing the client version with the latest update version.

4. A method as recited in claim 2 wherein the version of the client copy of the database table is referenced from client profile information and the determination the necessary updates is done by comparing the client version with the latest update version.

5. A method as recited in claim 2 wherein each update contains the database differences in a generic format and further comprises the steps of:

ascertaining the client database type wherein the client copy of the database table is held; and

determining which updates are necessary for making the client copy of the database table current; and

generating the database differences based upon the necessary updates prior to translation into instructions.

15. A method as recited in claim 12 wherein the client computer supplies the version of the client copy of the database table as part of the request and the determination the necessary updates is done by comparing the client version with the latest update version.

16. A method as recited in claim 12 wherein the version of the client copy of the database table is referenced from client profile information and the determination the necessary updates is done by comparing the client version with the latest update version.

17. A computer-readable medium having computer-executable instructions for performing the steps recited in claim 9.

18. A method as recited in claim 9 wherein the client is intermittently connected to the server computer.

19. A computer program product comprising:

a computer usable medium having computer readable program code means embodied in said medium for distributing database differences corresponding to database change events made to a database table located on a server computer to client copies of the database table located on one or more client computers, each client computer capable of having one or more database engines, said computer readable program code means comprising;

means for creating and storing on the server computer one or more sequentially versioned updates, each update containing database differences corresponding to database change events made to the database table since the preceding update;

means for receiving from a client computer a request for all the database differences needed to make the client copy of the database table current;

means for determining which updates are necessary for making the client copy of the database table current;

means for ascertaining the client database type associated with the client copy of the database table;

means for translating the specific differences based upon the necessary updates from a generic format into instructions specific to the client database type; and

means, electronically coupled and responsive to said server CPU, for transmitting the instructions to the client computer for execution on the client database engine to make the client copy of the database table current; and

c) at least one client computer system comprising:

a client CPU;

client storage means, electronically coupled and responsive to said client CPU, wherein said client storage means is partitioned into at least a first memory partition for storing the client copy of a database table;

interface means, electronically coupled and responsive to said client CPU, for establishing an intermittent means for electronic communication to said communications network;

means, electronically coupled and responsive to said client CPU, for sending to said server computer a request for all the database differences needed to make the client copy of the database table current;

means, electronically coupled and responsive to said client CPU, for receiving from said server computer instructions for making the client copy of the database table current; and

means, electronically coupled and responsive to said client CPU, for applying said instructions to the client copy of database table thereby making it current.

NAME	LETTER GREETING	EMPLOYEE NUMBER	MARITAL STATUS
JONES, G.J.	MR. JONES	7843	SINGLE
PRESLEY, E.	MR. PRESLEY	8782	MARRIED
SMITH, T.O.	MRS. SMITH	1703	MARRIED
WRIGHT, H.D.	MISS WRIGHT	0813	SINGLE

FIG. 2A

NAME	LETTER GREETING	EMPLOYEE NUMBER	MARITAL STATUS
JONES, G.J.	MR. JONES	7843	MARRIED
MAUSS, B.	MR. MAUSS	7777	SINGLE
SMITH, T.O.	MRS. SMITH	1703	MARRIED
WRIGHT, H.D.	MISS WRIGHT	0813	SINGLE

FIG. 2B

NAME	LETTER GREETING	EMPLOYEE NUMBER	MARITAL STATUS
JONES, G.J.	MR. JONES	7843	MARRIED
MAUSS, B.	MR. MAUSS	7777	SINGLE
SMITH, T.O.	MRS. SMITH	1703	MARRIED
YOUNG, H.D.	MRS. YOUNG	0813	MARRIED

FIG. 2C

NAME	TITLE	LETTER GREETING	EMPLOYEE NUMBER	MARITAL STATUS
JONES, G.J.	ASSOCIATE ATTORNEY	MR. JONES	7843	MARRIED
MAUSS, B.	PRESIDENT	MR. MAUSS	7777	SINGLE
SMITH, T.O.	NURSE	MRS. SMITH	1703	MARRIED
YOUNG, H.D.	OFFICE MANAGER	MRS. YOUNG	0813	MARRIED

FIG. 2D

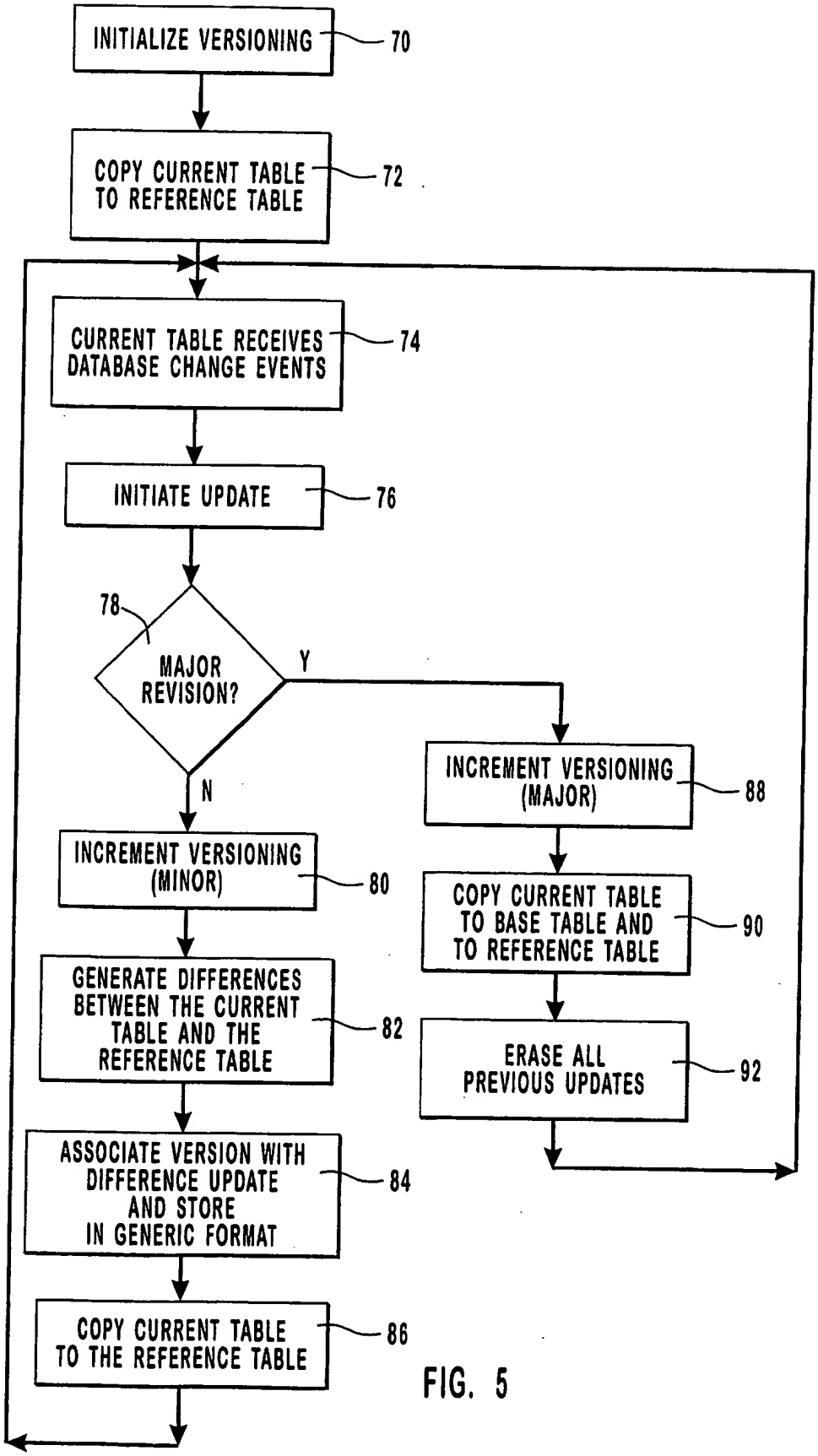


FIG. 5

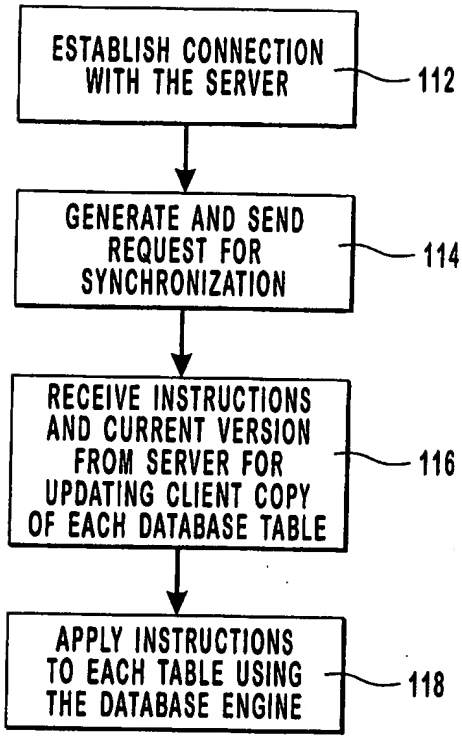


FIG. 7